What is Claimed is:

1. A method of circuit equivalence checking, comprising:

solving at least a first equivalence checking problem;

storing at least a first solution to at least the first equivalence checking problem; and

reusing at least part of the first solution to the first equivalence checking problem for a second equivalence checking problem.

2. The method of claim 1, wherein at least one of the first equivalence checking problem and the second equivalence checking problem require determination of logical equivalence between at least combinational circuits.

3. The method of claim 1, wherein at least one of the first equivalence checking problem and the second equivalence checking problem require determination of logical equivalence between at least sequential logic circuits.

4. The method of claim 1, wherein the first solution includes one or more problem signatures.

5. The method of claim 4, wherein the one or more problem signatures include pre-cached problem signatures.

6. The method of claim 5, wherein the pre-cached problem signatures includes one or more numbers of inputs.

7. The method of claim 5, wherein the pre-cached problem signatures includes one or more numbers of intermediate signals.

8. The method of claim 5, wherein the pre-cached problem signatures includes one or more counts of logic gates.

9. The method of claim 8, wherein the pre-cached problem signatures includes one or more counts of logic gates of different types.

10. The method of claim 5, wherein the pre-cached problem signatures includes RTL information.

11. The method of claim 5, wherein the pre-cached problem signatures includes hierarchical information.

12. The method of claim 4, wherein the one or more problem signatures include cached problem signatures.

13. The method of claim 12, wherein the cached problem signatures includes one or more numbers of inputs.

14. The method of claim 12, wherein the cached problem signatures includes one or more numbers of intermediate signals.

15. The method of claim 12, wherein the cached problem signatures includes one or more counts of logic gates.

16. The method of claim 15, wherein the cached problem signatures includes one or more counts of logic gates of different types.

17. The method of claim 12, wherein the cached problem signatures includes RTL information.

18. The method of claim 12, wherein the cached problem signatures includes hierarchical information.

19. The method of claim 1, wherein the first solution includes one or more algorithm traces.

20. The method of claim 1, wherein the first solution includes one or more stored problems.

21. The method of claim 1, wherein the first equivalence checking problem includes comparing logic cones.

22. The method of claim 21, wherein the first solution includes one or more problem signatures.

23. The method of claim 22, wherein the one or more problem signatures includes previously compared logic cones.

24. The method of claim 22, wherein the one or more problem signatures includes hash values of previously compared logic cones.

25. The method of claim 22, wherein the one or more problem signatures includes counts of primary inputs and primary outputs.

26. The method of claim 22, wherein the one or more problem signatures includes counts of logic gates.

27. The method of claim 26, wherein the one or more problem signatures includes counts of logic gates of different logic gate type.

28. The method of claim 22, wherein the one or more problem signatures includes quasi canonical numbers associated with graph colorings of compared logic cones.

29. The method of claim 22, wherein the one or more problem signatures includes RTL expressions.

30. The method of claim 22, wherein the one or more problem signatures includes RTL key point type counts.

31. The method of claim 21, wherein the one or more problem signatures includes RTL key point types.

32. The method of claim 21, wherein the first solution includes one or more algorithm traces.

33. The method of claim 32, wherein the one or more algorithm traces includes a list of algorithms used for comparing logic cones.

34. The method of claim 32, wherein the one or more algorithm traces includes an order of algorithms used for comparing logic cones.

35. The method of claim 32, wherein the one or more algorithm traces includes one or more intermediate values of algorithms used for comparing logic cones.

36. The method of claim 32, wherein the one or more algorithm traces includes one or more final results of algorithms used for comparing logic cones.

37. The method of claim 32, wherein the one or more algorithm traces includes one or more don't care models.

38. The method of claim 32, wherein the one or more algorithm traces includes one or more phases.

39. The method of claim 32, wherein the one or more algorithm traces includes one or more attempted algorithms.

40. The method of claim 32, wherein the one or more algorithm traces includes resource limitation data on one or more attempted algorithms.

41. The method of claim 32, wherein the one or more algorithm traces includes one or more successful algorithms.

42. The method of claim 32, wherein the one or more algorithm traces includes limitation data on one or more successful algorithms.

43. The method of claim 21, wherein the first solution includes one or more stored problems.

44. The method of claim 43, wherein the one or more stored problems permit exact matching between stored and searched logic cone pairs.

45. The method of claim 1, wherein the first equivalence checking problem includes determining one or more sensitizing simulation vectors.

46. The method of claim 45, wherein the first solution includes one or more problem signatures.

47. The method of claim 46, wherein the one or more problem signatures includes logic cones sensitized by the one or more sensitizing simulation vectors.

48. The method of claim 46, wherein the one or more problem signatures includes hash values of the logic cones sensitized by the one or more sensitizing simulation vectors.

49. The method of claim 45, wherein the first solution includes one or more algorithm traces.

50. The method of claim 49, wherein the one or more algorithm traces includes the one or more sensitizing simulation vectors.

51. The method of claim 1, wherein the first equivalence checking problem includes mapping state elements.

52. The method of claim 51, wherein the first solution includes one or more problem signatures.

53. The method of claim 52, wherein the one or more problem signatures includes one or more state point pairs.

54. The method of claim 52, wherein the one or more problem signatures includes one or more supporting logic cones.

55. The method of claim 52, wherein the one or more problem signatures includes one or more signatures of supporting logic cones.

56. The method of claim 52, wherein the one or more problem signatures includes one or more driving logic cones.

57. The method of claim 52, wherein the one or more problem signatures includes one or more signatures of driving logic cones.

58. The method of claim 51, wherein the first solution includes one or more algorithm traces.

59. The method of claim 51, wherein the one or more algorithm traces includes one or more mapping algorithms that mapped one or more state point pairs of the first solution.

60. The method of claim 51, further comprising:

extracting transitive properties to combine at least two solutions to at least one equivalence checking problem.

61. The method of claim 60, further comprising:

predetermining at least one solution to at least one equivalence checking problem, using the at least two solutions.

62. The method of claim 60, wherein at least one of the transitive properties is a correspondence of intermediate signals between two compared designs.

63. An apparatus for circuit equivalence checking, comprising:

one or more persistent caches including:

one or more cached objects including:

one or more problem signatures; and

one or more algorithm traces; and

one or more cache managers accepting one or more equivalence checking queries and returning at least part of at least one cached object from the one or more persistent caches at least partly in response to the one or more equivalence checking queries.

64. The method of claim 63, wherein the one or more persistent caches is populated with at least one cached object prior to the one or more cache managers accepting one or more equivalence checking queries.

65. The method of claim 63, wherein at least partly in response to the one or more equivalence checking queries, the one or more persistent caches is updated.

66. The method of claim 65, wherein the one or more persistent caches is updated at least by adding at least one cached object to the one or more persistent caches.

67. The method of claim 65, wherein the one or more persistent caches is updated at least by removing at least one cached object from the one or more persistent caches.

68. The method of claim 65, wherein the one or more persistent caches is updated at least by changing at least one cached object in the one or more persistent caches.

69. The method of claim 65, wherein the one or more persistent caches is updated at least by replacing at least one cached object in the one or more persistent caches.

70. The method of claim 63, wherein the one or more persistent caches are searched with low cost algorithms prior to high cost algorithms.